U.S. Department of Energy Office of Science

Solicitation LAB01-01

National Collaboratories and High Performance Networks

Title of Proposed Project:

The Distributed Monitoring Framework (DMF)

Name of Laboratory:	Lawrence Berkeley National Laboratory
Principal investigator:	Brian L. Tierney
Position title of PI:	Staff Scientist, Lawrence Berkeley National Laboratory
Mailing Address:	1 Cyclotron Rd. 50B-2239
Phone:	510/486-7381
FAX:	510/486-6363
Email:	bltierney@lbl.gov

A proposal for the period October 1, 2001 – September 30, 2004

(Note: the orginal proposal was only funded at 50%, so this is a revised version that includes only the funded components.)

Abstract

The goal of the Distributed Monitoring Framework is to improve end-to-end data throughput for data intensive applications in a high-speed WAN environments, and to provide the ability to do performance analysis and fault detection in a Grid computing environment. This monitoring framework will provide accurate, detailed, and adaptive monitoring of all of distributed computing components, including the network. Analysis tools will be able to use this monitoring data for real-time analysis, anomaly identification, and response. Services for network-aware applications will use this monitoring infrastructure in order to provide past, present, and future predictions of the network conditions, which the applications can then use to adapt their behavior. Applications such as widely distributed workflow management systems can then use the DMF as a generalized real-time event service.

Access to timely and accurate performance monitoring data is critical for enabling high performance data intensive computing. A example of an application area requiring this monitoring data is distributed data location management, which uses host, network, and middleware monitoring data in order to select an optimal data replica. For several years the Data Intensive Distributed Computing group has been working on tools for monitoring and analyzing distributed system performance. These tools have proved to be very successful, enabling us to win the SC2000 Networking Bandwidth Challenge with an application capable of using 1.5 Gb/s across a WAN. In particular our NetLogger Toolkit is very popular, and is being used by many large projects, including Globus and the EU DataGrid.

Using common protocols and data formats we will unify our monitoring tools into a framework we call the Distributed Monitoring Framework (DMF). This framework is composed of the following components:

- sensors to monitor every aspect of the distributed system (networks, disks, hosts, applications and middleware)
- a sensor management system, to control the sensors
- an event publishing and archiving system, to allow event consumers to locate current and past event data
- a common set of protocols for exchanging and locating monitoring data

This program has the following research objectives:

- Integration and enhancement of existing monitoring tools and technologies into a Grid environment
- Development of non-intrusive methods for network capacity monitoring and estimation
- Determination of what network metrics are needed by network-aware DOE Science Grid applications

Deliverables from this work will include:

- Design, implementation, and evaluation of a scalable, fault tolerant monitoring and event infrastructure
- Deployment in PPDG, the EU DataGrid, and the DOE Science Grid projects

1.0 Proposal Narrative

1.1 Background and Significance

The next generation of high-speed networks will allow DOE scientists unprecedented levels of collaboration. A key element of this collaboration will be transparent access to large data archives from anywhere on the network. During the past few years the Data Intensive Distributed Computing (DIDC) Group at LBNL has been focused on techniques to improve wide-area throughput for data intensive, or "Data Grid", applications. To this end, we developed a number of tools and services for instrumentation and monitoring. These include the NetLogger Toolkit, the Enable network monitoring service, the Grid Monitoring Architecture (GMA), and others. These tools have proved to be very successful, enabling a team from LBNL to win the IEEE Supercomputing 2000 Networking Bandwidth Challenge with an application capable of using 1.5 Gb/s across a WAN.

The focus of this proposal is to extend our previous work on monitoring and instrumentation, better integrating it into a Grid environment [13], and to determine what types of monitoring data is most useful to Data Grid middleware.

1.1.1 Distributed Monitoring Framework

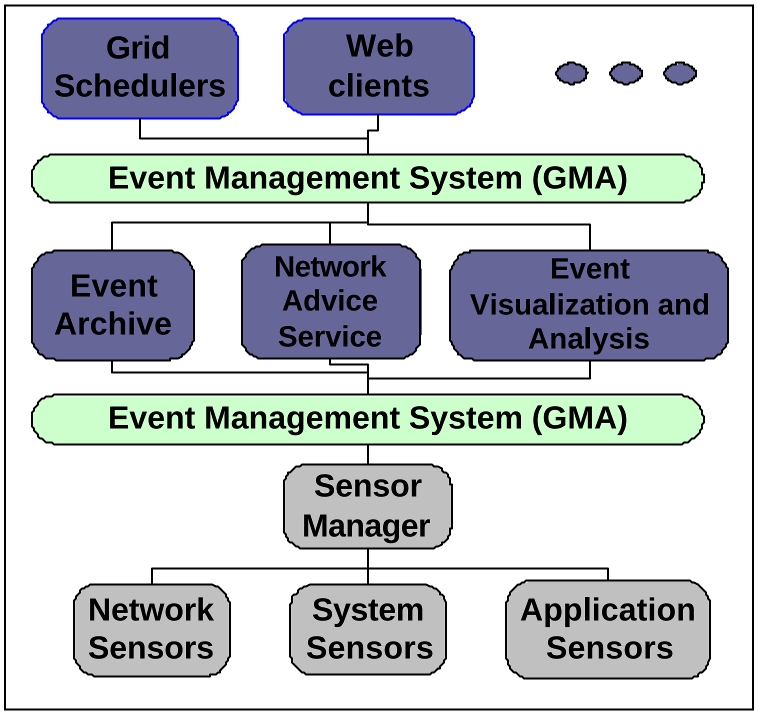

Figure 1: The Distributed Monitoring Framework

This proposed project will develop and deploy an integrated set of services that we are calling the Distributed Monitoring Framework (DMF). The Distributed Monitoring Framework is composed of the following components, shown in Figure1. At the lowest level there are several types of sensors to monitor every aspect of the distributed system, including networks, disks, hosts, applications, and middleware. At the next level, a sensor management system provides a secure mechanism for controlling and managing the sensors. A common protocol for accessing the sensor data is provided by the Grid Monitoring Architecture (GMA) implementation layer. The measurements provided by the sensors are grouped into "*events*". Precision instrumentation of applications and middleware will also be provided as *events* by GMA. Clients of the DMF are consumers of the monitoring event data. These include such services as network tuning advice services (e.g.: tell me what TCP buffer to use for a given link), and event visualization and analysis tools for debugging and performance tuning. Event data needs to be aggregated and stored for later analysis; this function will be performed by a real-time event archive that acts as a DMF consumer, and can be, in turn, queried through the GMA Producer interface. The GMA interface allows filters or archives to be layered on top of each other, thus clients can use the same protocol to obtain data from producers at any level above the sensors themselves. For example, clients such as Grid schedulers (sometimes called "superschedulers" [33]), can use monitoring data from either the real-time archive or directly from the Sensor Management system, to determine the optimal combination of computer and data resources to run a given Grid job.

Many of the components of the DMF have already been prototyped or implemented by the DIDC Group. Each of these is described in some detail below, but a brief summary is presented here. The NetLogger Toolkit [25] includes application sensors, some system and network sensors, a powerful event visualization tool, and a simple event archive. The DIDC "Network characterization Service" [28] has proven to be a very useful hop-by-hop network sensor. Our work on the Global Grid Forum Grid Monitoring Architecture (GMA) addressed the event management system [39]. Our JAMM (Java Agents for Monitoring Management) is preliminary work on sensor management [42]. The Enable project produced a simple network tuning advice service [41].

Our proposed DMF will integrate and enhance these complementary components, and ensure they all scale for use in a Grid environment. The GMA will be used to unify the various sensor output formats and publication mechanisms into a single, standardized view for the Grid. The GMA has proven to be an elegant concept, but it has yet to be tested in a real Grid environment.

Large Grid projects such as the Particle Physics Data Grid, the EU DataGrid, and the DOE Science Grid will require such as monitoring framework if they are to be successful. In particular, data intensive Grid projects, such as the High Energy Physics Grid projects, which will handle hundreds of terabytes of data, require efficient use of networking that can only be provided by such a monitoring framework. We have been working closely with the EU DataGrid Project, and propose to work with the PPDG and DOE Science Grid projects as well to deploy and test the DMF in these environments.

1.2 Proposal Overview

There are three main components to this proposal. The first component is the Distributed Monitoring Framework, a comprehensive framework generating, publishing, and archiving monitor event data for Grid applications and middleware. The second component is the integration of these components into Grid projects such as PPDG, EU DataGrid, and the DOE Science Grid.

1.2.1 Distributed Monitoring Framework

Our planned work with the DMF will consist of integrating and extending an existing set of complementary components, followed by evaluation, testing and refinement. A key role of the DMF will be to provide a unifying view to a wide range of sensor data, from network to host to application. Standard event descriptions, data formats, and publish/subscribe APIs and protocols are all required. The main components of the DMF are *instrumentation*, *sensors*, *sensor management, event publication*, and *event archiving*.

Instrumentation

The ability to do precision, real-time instrumentation of Grid applications and middleware is essential to the process of developing high performance data intensive applications. DMF will include tools to make it easy to non-intrusively add instrumentation to Grid middleware, and to publish this event data in a standard manner. Our previous work on the NetLogger Toolkit will provide the basis for this component.

Sensors

Network and host sensors, combined with instrumented applications, allow one to do end-to-end performance analysis. The DMF will define standard schemas and publication mechanisms for this sensor data. In particular, we will focus on network sensors. Based on our work on the “Network Characterization Service”, described below, the DMF will include a non-intrusive network sensors capable of hop-by-hop network analysis. There are plenty of existing host sensors available, and a subset of these will be integrated into the DMF. Other network monitoring work such as the Net100 project [24], and the SciDAC “Self Configuring Network Monitor” project [32], whose goal is to design and deploy a passive monitoring infrastructure, will also be integrated.

Sensor Management System

As distributed systems become bigger and more complex, there are more pieces to monitor and manage. Some components will require constant monitoring, while others will only be monitored on demand. The sensors themselves need to be automatically installed, updated, and removed. We are developing a Sensor Management System to securely control the distribution and execution of monitoring sensors in a distributed/Grid environment.

Event Publication

To handle the potentially huge amounts of sensor event data requires a flexible, highly scalable event publication and subscription service. The Grid Monitoring Architecture (GMA), described below, is designed for this purpose. Several GMA implementation have started to appear, but so far these are largely untested. We propose to deploy GMA in the DOE Science Grid and other testbeds to determine the scalability and interoperability of the GMA. In addition, a great deal of work remains to define standard event schemas and event dictionaries for the GMA. We also plan to explore the use of GMA as a generalized event handling system for Grid environments.

Event Archives

The ability to archive event data is critical for performance analysis and tuning, as well as for accounting purposes. The archive must be extremely high performance and scalable to ensure that it does not become a bottleneck. The GMA defines an archive to be a special type of consumer, as well as a producer, but this model has not yet been implemented or tested. We propose to extend the simple NetLogger mySQL-based archive, integrating it into GMA, and designing a query interface for the archive.

1.2.2 Clients of the DMF

To validate the DMF approach, we propose to design and develop two Grid services that depend on the DMF.

Network Advice Service

Making sense out of network sensor data typically requires a great deal of network expertise. We have previously developed a network advice service, called Enable (described below), that makes it easy for application developers to obtain access to the network sensor data that is most useful to them, without becoming a network expert. We propose to use Enable as a test client for the DMF, consuming network event data from GMA using standard event formats, and logging all Enable results to the DMF archive.

1.2.3 Integration with other Grid Projects

In order for Grid projects such as PPDG, the EU DataGrid, and the DOE Science Grid to be successful, there must exist a flexible, scalable, monitoring framework that provide standard APIs and protocols. Currently most Grid application projects are using the Globus Metadata Directory Service (MDS) [12] to provide monitoring capabilities. However MDS was designed for relatively static data, such as host operating system or memory size, and only later was it extended for dynamic monitoring information. We believe the GMA model, which was designed primarily for highly dynamic environments, is much better suited to form the basis of a monitoring framework. Also, MDS uses LDAP for it’s protocol and API, while GMA is using XML, providing many additional options for leveraging existing and future web services.

We currently have very close ties with the EU DataGrid project, and will work with them to define requirements for the DMF, and to test DMF components in this environment. We propose to work with the PPDG and DOE Science Grid projects as well, and deploy and test the DMF for use by these projects. Another important role we will play is to attempt to insure interoperability between the US and EU Grid project’s monitoring systems.

1.2.4 Experience and Competence

The LBNL Distributed Systems Department has a great deal of experience in building and tuning data-intensive, wide-area distributed applications. The NetLogger Toolkit, described below, was developed in order to tune and debug distributed applications, has been used extensively in several high-speed networking applications, including the DPSS [40], Radiance, BaBAR [38], Visapult [2], and GridFTP [20]. This experience has made us highly aware of functionality gaps in the currently available monitoring facilities, and the difficulties of getting the applications and the operating systems to fully utilize high-bandwidth network links.

At the Supercomputing 2000 conference an LBNL team received the “Fastest and Fattest” Network Challenge award for the application that best utilized the wide-area network. The application was a remote data visualization application called Visapult [2], which used a peak rate of 1.5 Gbits/second, and a sustained data rate of 680Mbps. This performance was the result of a great deal of hand tuning of the SC2000 network. Visapult transforms data from a simulation using parallel compute nodes and transmits the transformed data in parallel over the network for rendering. The dataset, 80 GB in size, was stored at LBNL and the compute cluster, an 8-processor SGI Origin with 4 Gigabit Ethernet interfaces, was in Dallas, TX. Parallel reading of large datasets stored at a distant location is common in high-energy physics (HEP) applications.

LBNL has also been heavily involved in the Global Grid Forum (GGF) [14] and, of particular interest here, the GF Performance Working Group. LBNL co-authored the white paper for a monitoring data architecture called the Grid Monitoring Architecture (GMA) [39]. Through our GGF work we have developed a thorough knowledge of the issues involved in designing and a distributed monitoring system. The GMA design was heavily influenced by our experiences developing a system called Java Agents for Monitoring and Management (JAMM) [42].

1.3 Preliminary Studies

The components for the Distributed Monitoring Framework are based on the following past DIDC projects: NetLogger, Enable, GMA, JAMM, and NCSD. We now describe each of these.

1.3.1 NetLogger Toolkit

The LBNL *NetLogger Toolkit* [25] is designed to monitor, under actual operating conditions, the behavior of all the elements of the application-to-application communication path in order to determine exactly where time is spent within a complex system. Our experience has shown that often distributed application developers blame the network for their performance problems, when in fact the problem is application design. Often better pipelining of I/O and computation by the application will lead to dramatically increased performance.

Using NetLogger, distributed application components are modified to produce timestamped logs of "interesting" events at all the critical points of the distributed system. The events are correlated with the system's behavior in order to characterize the performance of all aspects of the system and network in detail.

All the tools in the NetLogger Toolkit share a common log format, and assume the existence of accurate and synchronized system clocks. The NetLogger Toolkit itself consists of four components: an API and library of functions to simplify the generation of application-level event logs, a set of tools for collecting and sorting log files, an event archive system, and a tool for visualization and analysis of the log files. In order to instrument an application to produce event logs, the application developer inserts calls to the NetLogger API at all the critical points in the code, then links the application with the NetLogger library.

We have found exploratory, visual analysis of the log event data to be the most useful means of determining the causes of performance anomalies. The NetLogger Visualization tool, *nlv*, has been developed to provide a flexible and interactive graphical representation of system-level and application-level events.

Figure 1 shows sample *nlv* results, using a remote data copy application. The events being monitored are shown on the Y axis, and time is on the X axis. Each *lifeline* represents one block of data, and one can easily see that most of the time is spent between ST_SEND_DATA and END_SEND_DATA, which is the network data transfer time. At the bottom of the figure there is a scatter plot showing the size of the data blocks that arrived at the receiving application.

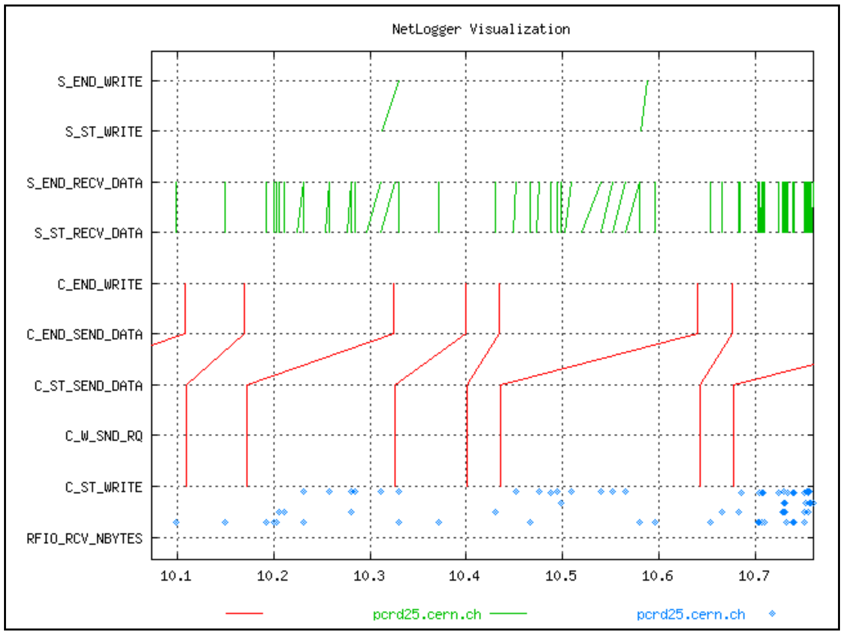

Figure 1 Sample NetLogger Results

1.3.2 Grid Monitoring Architecture (GMA)

Large distributed systems such as Computational and Data Grids require a substantial amount of monitoring data be collected for a variety of tasks such as fault detection, performance analysis, performance tuning, performance prediction, and scheduling. A Grid monitoring system is differentiated from a general monitoring system in that it must be scalable across wide-area networks, and include a wide range of heterogeneous resources. It must also be integrated with other Grid middleware in terms of naming and security issues. The DIDC group has led a Global Grid Forum (GGF) effort to defined a highly scalable *Grid Monitoring Architecture*, or GMA. The GGF Performance Working Group is also working to standardize the protocols and architectures for the management of a wide range of Grid monitoring information, including network monitoring.

In some models, such as the CORBA Event Service [6], all communication flows through a central component, which represents a potential bottleneck. In contrast, GMA performance monitoring event data travels directly from the producers of the data to the consumers of the data. In this way, individual producer/consumer pairs can do "impedance matching" based on negotiated requirements, and the amount of data flowing through the system can be controlled in a precise and localized fashion. The design also allows for replication and reduction of event data at intermediate components acting as caches or filters.

Terminology

The monitoring data that the GMA is designed to handle are timestamped ***events***. An event is a named structure that may contain one or more items of data that relate to one or more resources such as memory usage or network usage, or application-specific types of data like the amount of time it took to multiply two matrices. The *producer* (shown in Figure2) is the component that makes the event data available. A *consumer* is any process that requests or accepts event data. A *directory service* is used to publish what event data is available and which producer to contact to request it.

The GMA architecture supports both a publish/subscribe model and a query/response model. In the first case, event data is streamed over a "channel" that is established with an initial request. For query/response, one item of event data is returned per request, similar to a remote procedure call (RPC).

By defining three interfaces -- the consumer to producer interface, the consumer to directory service interface, and the producer to directory service interface -- diverse Grid monitoring services can interoperate.

The **directory service** contains only metadata about the performance events, and a mapping to their associated producers or consumers. In order to deliver high volumes of data an scale to many producers and consumers, the directory service is not responsible for the storage of event data itself.

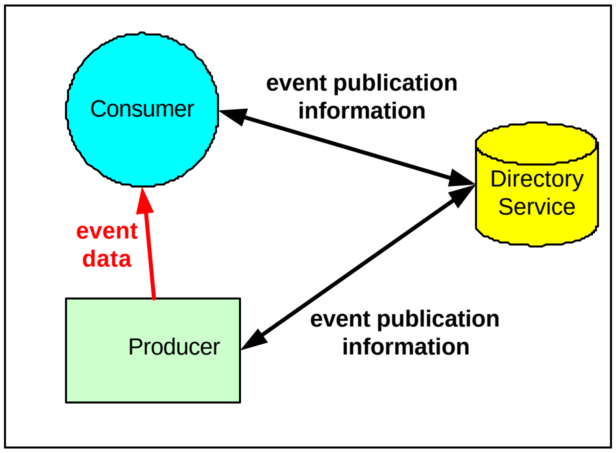

Figure 2: Grid Monitoring Architecture Components

A **consumer** is any program that requests and/or receives event data from a producer. In order to find a producer that provides desired events, the consumer can search the directory service. A consumer which passively accepts event data from a producer may register itself, and what events it is willing to accept, in the directory service.

A **producer** is an program that responds to consumer requests and/or sends event data to a consumer. A producer that accepts requests for event data will register itself and the events it is willing to provide in the directory service. In order to find a consumer that will accept events that it wishes to send, a producer can search the directory service.

There may be components that are both consumers and producers, called "**consumer/producer pipes**". For example a consumer might collect event data from several producers, and then use that data to generate a new derived event data type, which is then made available to other consumers, as shown in Figure3. More elaborate filtering, forwarding, and caching behaviors could be implemented by connecting multiple consumer/producer pipes.

GMA has proven to be an elegant design, and several group are currently working on GMA implementations. However a great deal of work remains to be done. This work is outlined in section 1.5.1 below.

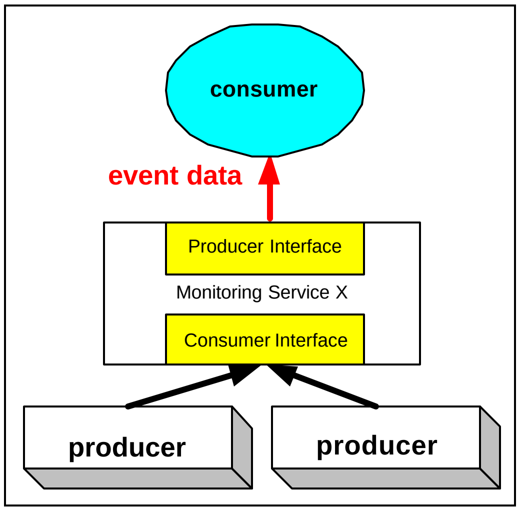

Figure 3: GMA Consumer/Producer Pipes

1.3.3 Enable Network Monitoring Service

There exists a large body of work showing that good network performance can be achieved using the proper tuning techniques. The most important technique is the use of the optimal TCP buffer size, and techniques for determining the optimal value for the TCP buffer size are described in [44]. Another important technique is to use parallel sockets, as described in [35]. However, determining the correct tuning parameters can be quite difficult, especially for users or developers who are not network experts. The optimal TCP buffer size and number of parallel streams are different for every network path, vary over time, and vary depending on the configuration of the end hosts. There are several tools that help determine these values, such as *iperf* [19], *pchar* [27], and *pipechar* [18], but none of these include a client API, and all require some level of network expertise to use. Other groups are addressing this problem at the kernel level, such as the web100 project [45], Linux 2.4 [21], and others [11].

The DIDC group has designed and developed a service which provides clients with the correct tuning parameters for a given network path. We call this service ***Enable***, because it enables applications to optimize their use of the network and achieve the highest possible throughput. The goal of the Enable service is to eliminate what has been called the *wizard gap* [22]. The wizard gap is the difference between the network performance that a network "wizard" can achieve by doing the proper tuning, compared to the performance of an untuned application. The Enable service can act as that wizard.

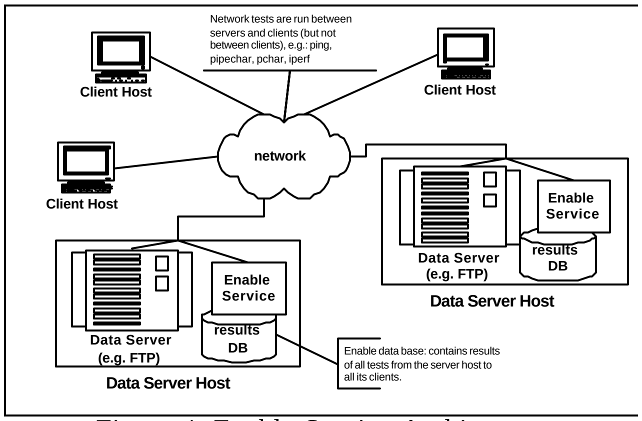
Network tests are run between servers and clients (but not between clients), e.g.: ping, pipechar, pchar, iperf
Client Host
Client Host
Client Host
network
Enable Service
results DB
Data Server (e.g. FTP)
Data Server Host
Enable Service
results DB
Data Server (e.g. FTP)
Data Server Host
Enable data base: contains results of all tests from the server host to all its clients.

Figure 4: Enable Service Architecture

From the application developer's perspective, Enable provides advice on the correct tuning parameters without requiring knowledge about how these are obtained. Thus, the selected algorithms and tools for computing these parameters can be changed transparently to the application. The network tuning parameters that the Enable service is initially concentrating on are those required by large bulk data transfer applications, such as the various "Data Grid" [4] projects.

The Enable service works as follows: An Enable server is co-located on every system that is serving large data files to the wide-area network (e.g.: an FTP or HTTP server). The Enable service is then configured to monitor the network links to a set of client hosts from the perspective of that data server. Network monitoring results are stored in a database, and can be queried by network-aware distributed components at any time. The Enable service runs the network tests on some pre-configured time interval, e.g., every 6 hours or whenever a new client connects. Enable includes a simple client API that makes querying the Enable Servers trivial for application developers. A sample deployment of Enable is shown in Figure4.

1.3.4 Network Characterization Service (pipechar and ncsd)

Grid middleware needs an accurate estimate of network bandwidth in order to perform a number of tasks, including network tuning, query optimization, scheduling, job execution time estimation, replica selection, and so on. Additionally, hop-by-hop network analysis is increasingly critical to network troubleshooting on the rapidly growing Internet. The Network Characterization Service (NCS) provides the ability find bottleneck hops and to diagnose and troubleshoot networks hop-by-hop in an easy and timely fashion.

The tool ***pipechar*** [28], developed by the DIDC group, is used for characterization of network problems via hop-by-hop bandwidth measurement. Unlike tools such as *iperf* and *ttcp*, pipechar makes all its measurements by sending packets from a single host, a technique called "sender-only" bandwidth measurement. This means that you only need a login on one end of a connection to test the end-to-end bandwidth. There are other tools that employ this method, including *pchar* [27]. However, whereas *pchar* may take 30-40 minutes to measure a typical WAN paths, *pipechar* usually takes on the order of a few minutes. More importantly, *pipechar* is unique among sender-only tools in its ability to measure high-speed links (other tools are limited to about 100Mb/s).

An additional advantage of the methods used by *pipechar* is that the test itself uses only a small amount of a high-speed network's bandwidth, thus unlike active bandwidth tests such as *iperf*, pipechar will not strongly interfere with actual application traffic. For example, recent testing from CERN to LBNL shows that to achieve good average throughput result from *iperf* requires a 10 minute test, which would clearly be intrusive to other applications running at the time.

While *pipechar* is a command-line tool, it was designed to be used in conjunction with LBNL's Network Characterization Service (NCS) [18]. In order to reduce redundant testing from the same subnet or through the same bottleneck router, the Network Characterization Service has been developed to stores bandwidth measurements made with pipechar and responds to bandwidth queries with cached results of each hop in the path, minimizing the amount of testing necessary to keep an up-to-date view of network performance.

To be useful with the Distributed Monitoring Framework, schemas for NCS results must be designed, and NCS must be modified to support the GMA producer interface.

1.3.5 Java Agents for Monitoring and Management

In complex distributed systems, installing sensors, configuring these sensors to send their data to interested parties, and dynamically starting and stopping monitoring based on system activity is a difficult task for a single administrator. In the Grid environment, which by its nature crosses administrative domains, these tasks must be automated. The DIDC group developed a system called Java Agents for Monitoring and Management (JAMM) to address these problems. The JAMM architecture, includes a distributed set of components which collect and publish monitoring data about computer systems. Clients may either control the execution of sensors, or they may receive the events from one or more sensors. The similarity between JAMM and the GMA architecture is strong because the design principles used for JAMM were a key input to the design of the GMA.

In addition to allowing centralized control of sensors using a Java GUI, JAMM also has a component called the *portmonitor*, that can be configured to start and stop sensor activity automatically based on port activity on a host. Because distributed applications often use pre-defined ports or network services, this has proved a very useful mechanism for providing automatic and fine-grained control of application monitoring.

1.4 Other Related Projects

There are many projects related to one of more of the DMF components, however we believe there are no projects that integrate all of the components into a single framework. Some of the more important related projects are the Network Weather Service [46], Remos [7], Pablo [31], CODE [36], and the Globus MDS [12]. Each of these projects uses its own protocols and architecture, so therefore they cannot interoperate without a mediating layer. One of our goals for the DMF is to provide a Global Grid Forum standard interface to facilitate integrating diverse systems such as those just mentioned.

In addition, our experience with high volumes of event data from NetLogger application instrumentation has led us to design our systems to scale well to hundreds or thousands of event per second, whereas many monitoring systems such as the Globus MDS and Pablo from UIUC, are designed to handle a relatively small amount of monitoring data. The monitoring framework we are developing, based on NetLogger and GMA, is unique in its ability to handle large volumes of monitoring data without affecting application performance.

Another related system is the Network Weather Service (NWS). NWS includes a nice framework for collecting and publishing sensor event data, and NWS designers contributed to the GMA design. However, our experiments have found NWS network sensor data to be quite inaccurate at predicting performance of Data Grid applications. We have much more expertise in TCP bulk data transfer issues, and we are confident that our network sensors will be much more useful to Data Grid applications.

1.5 Research Design and Methods

This section provides a description of the research, design and development of the main elements of the project. These are described from the "bottom up", that is from the measurements through the common protocols up to the event archival system.

1.5.1 Distributed Monitoring Framework

Instrumentation

The NetLogger Toolkit, described above, allows the developer to easily instrument a distributed application. However, in order to provide this instrumentation data to a GMA consumer, the data needs to be described in as GMA "events". We will accomplish this within the DMF by first implementing a translation layer on the event archiver, that can accept NetLogger data and store them as GMA events. At the same time, we will lead Global Grid Forum efforts to standardize an efficient (binary) format for GMA event data, and then implement this format as one of the output formats for the NetLogger Toolkit.

Sensors

There are many system and network sensors in existence today. We propose to identify a useful subset of these, and define "events" which describe their measurements, so that the Sensor Management System can publish their results to a Grid. In coordination with members of the Global Grid Forum, we are working on the following sub-tasks:

- determining the precision and accuracy of the sensor
- determining the usefulness of the sensor data to Grid middleware
- defining the event schema for the sensors

In particular, we will be focusing on network sensors. We are co-leading the new GGF "Network Measurements Working Group" [16], whose charter is to determine which network metrics are of use to Grid middleware and applications.

Sensor Management

The sensors and system probes which actively or passively measure the computing, storage, network, and application resources, must be securely and automatically distributed, updated, and controlled. In addition, the management system must be extremely fault-tolerant so that it does not fail under precisely the types of conditions it is trying to help debug. The Sensor Management system will build on work already done for JAMM, but will be refocused on the management of the sensors themselves. Thus, development of the Sensor Management System will focus on developing several components: an authentication and authorization component for sensor control, an internal "sensor status" system that can be configured to report problems and/or automatically restart failed sensors, and a "sensor repository" that allows secure distributed updates of scripts and binaries. In addition, Sensor Management will include tools to automatically trigger more monitoring when certain criteria are met, such as high traffic loads, high loss rates, or activity by specific applications. The use of the Globus *job manager* and Condor-G for starting and monitoring sensors will be explored.

Grid Monitoring Architecture (GMA)

The DIDC group has been designing an implementation of the GMA architecture called py-GMA, so named because it uses a high-level scripting language called Python. The design goals for this project are to provide a small, lightweight implementation that is efficient enough to handle thousands of events per second, as from application instrumentation, yet which can be easily deployed and run on heterogeneous platforms.

The py-GMA will support all the base functionality of the GMA, using the cross-platform Simple Object Access Protocol (SOAP) to communicate between the producer, consumer, and directory service components. The recently developed extensions to SOAP, which provide the security functionality of the Grid Security Infrastructure (GSI), will be used to provide authentication of requests. A significant advantage to the use of SOAP is the potential for integration with Web Services technologies such as WSDL and UDDI; where appropriate, these technologies can be used to provide monitoring data through Web-based Grid "portals", which are becoming an increasingly important interface for distributed scientific applications.

The py-GMA data "channel" for the publish/subscribe communication of events between the producer and consumer will use one of several formats native to NetLogger, including a highly efficient binary format and a highly readable text format. For interoperability with Grid systems, the published Simple XML Producer/Consumer event format will also be available.

This work includes the following research topics:

- Event caching: Applications will be interested in a small subset of the data. Cache services should be written which can hold "recent" data. More intelligence can be added to the caches to perform higher level services such as data reduction and filtering.
- Data format: A simple XML format has been proposed by the Grid Forum and is being implemented. Because XML is slow, not all needs will be fulfilled by this. Therefore a binary format will need to be developed.
- Security: Access control and authentication must be applied in a consistent way as a part of the GMA protocol, so that along each step of the way, the Directory Service, the Producer, or the Consumer (or Archiver), can be authenticated to and access control can be provided on the data. Grid security mechanisms such as the Globus GSI will be integrated.

Event Archive

An archival system can either be a consumer of performance event data, providing a repository for performance data, or a producer, providing the consumer with the requested event data. The archive will provide the ability to correlate data over a large time period and across levels of the network infrastructure. Standard protocols for querying the archive will be developed in cooperation with the Grid Forum Performance working group. [15]

As a consumer, the archive will receive events from one or more sensors, instrumented application components, or other Grid programs implementing the GMA producer interface. As a producer, the archive will process queries on the archived events and transparently access either real-time data or data from a database. Both the producer and consumer interfaces of the archive will advertise which events they provide or accept in a GMA directory service. The client can then search the directory service to find which archives contain the desired event data.

A diagram of the architecture is shown in Figure 5. Note that although in the diagram there is a 1:1 mapping between archives and GMA producers, the architecture allows more than one producer per archive and/or more than one archive per producer (i.e., an N:M mapping). Before discussing each component in detail, we will first step through a simple use-case of the architecture.

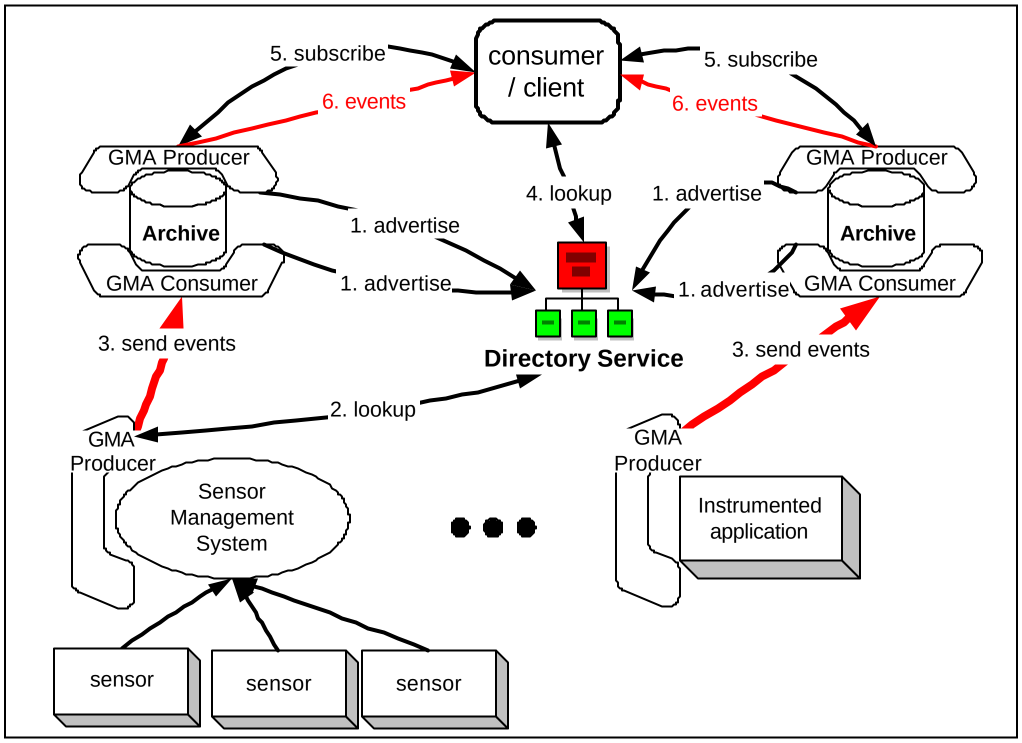

Figure 5: Event Archive

Archive Use-case

The use-case of a client querying the archive is illustrated below, with the numbers corresponding to annotations in Figure5.

1. The archives advertise the events they will accept, and from which set of producers, in the directory service. In this way, archives can pick which producers will "discover" them when the producers look for a consumer of their data (see step #2). They also advertise which set of events that they will provide to the consumer.
2. The GMA producer looks in the directory service to find a consumer (archive) that will accept its events. It then connects to the consumer interface of the archive; in the GMA, this is called a "producer-initiated subscribe".
3. Each producer starts sending its events to the appropriate archive.
4. The client wishing to do a query uses the directory service to locate the appropriate archives (which registered as a producer of those types of events in step #1).
5. The client queries, by using a parameterized "subscribe", the archives for events
6. The events are returned to the client from each archive in a stream.

Note that, 1) the directory service is not necessarily a single process but could be a hierarchy of meta-directory servers, and, 2) access control mechanisms are designed for all of these interfaces.

Archive Queries

We anticipate that most of the queries, at least initially, that would be asked of the system will have complexity similar to the query: "Retrieve events named (TCP_Connect) with parameters (source=*, dest=131.243.2) in time-range (1-Feb-2001 .. 6-Feb-2001)." This would search the archive for TCP connection requests from anywhere to the 131.243.2 subnet in a 5-day period.

Using the GMA protocols, the consumer would make this query by passing a *filter* as part of the subscribe request. Implementing this filter efficiently is an important area for research. Another area for research is the issue of querying multiple distributed archives. The obvious method is to simply construct a query that applies to the entire dataset and send it to each archive. This would work, but it is likely that it is more efficient to subdivide the query using information from the directory service. For example, if there are many different events desired, from several different subnets, but each archive advertises only one or two events and a few associated hosts, subdividing the query could dramatically reduce search time. Aggregation of the results can occur by simply "fanning out" the query to all the archives and combining the results in memory or on disk as they are returned. This may be perfectly adequate for the simple case when the archives hold non-overlapping data and the sequence of the returned data is not crucial. However, if either of those two conditions do not hold, a component will need to be developed which can perform a sort and join of incoming information.

1.6 Relationship to other projects

We plan to leverage our ties to several large "Data Grid" [4] projects to provide a realistic test environment for the tools and techniques developed in this proposal. These projects include he Particle Physics Data Grid [29], GriPhyN [17], the EU DataGrid [10], and the Earth Systems Grid [9]. These projects all require the efficient transfer of very large scientific data files across the network, and would all benefit from the work described here. We will also work closely with the SciDAC "DOE Science Grid" project, led by William Johnston, LBNL. This project proposes creating a multi-laboratory Collaboratory Pilot aimed at integrating, deploying, and supporting the persistent services needed for a scalable, robust, high-performance DOE Science Grid, thus creating the underpinnings for a DOE Science Grid Collaboratory Software Environment.

We will use our experience and expertise in tuning high-performance distributed applications to validate the techniques from this proposal. We also have close ties to ESNet, NTON [23], and SuperNet [37] networks, and will work with each of them to deploy these new network monitoring tools and services in each of these environments. This provides us with a very unique and important test environment for our research, and allows us to validate the utility of these tools and services in a real network environment.

We also plan to work closely with Arie Shoshani's Data Management Group at LBNL and to integrate DMF with their Storage Resource Manager (SRM). SRM includes an "estimation" API to provide clients with an estimate of how long it will take to copy a given file to the SRM. The DMF will provide the data to make this possible. This proposed work is part of the NERSC data intensive computing strategy, and is striving to coordinate the research being done in the Distributed Computing Department with other parts of NERSC. The goal of this is to build a computing architecture capable of handling the data intensive computing needs of users at NERSC in the years to come.

We are also part of the new “Self Configuring Network Monitor” and “Net100” projects. Both of these projects will provide valuable sources of network monitoring data, which can be published via the DMF. The Net100 project will also provide additional data in the form of operating system sensor data that can be closely coupled to application data.

These DMF will be integrated with the Globus Toolkit to become a standard Grid service, able to be used by any Globus client. We are working with Grid application developers to instrument their applications with NetLogger monitoring, and to make these applications network-aware using the DMF services.

We are very involved with the Global Grid Forum community, in particular we have been leading the “Grid Performance” working group, and have written or co-written a number of documents in this area.

1.7 Milestones

Year 1:

DMF

- completion of first GMA implementation
- binary NetLogger format for increased performance, and SOAP over GSI extensions to NetLogger
- event data archive system capable of handling large volumes of monitoring data
- integration of netarchd archiver and GMA
- development of web front end to event archive
- sensor management design and implementation
- continued GGF leadership role on defining GMA protocol and API standards
- continued GGF leadership role on defining network monitoring metrics for the Grid

Grid Project Integration

- work with PPDG and other Grid projects to define requirements for DMF
- deploy initial DMF on PPDG and DOE Science Grid testbeds
- help Grid application and middleware developers use NetLogger for performance analyses

Year 2:

DMF

- integration of DMF with Net100 sensors
- integration of DMF with Self Configuring Network Monitoring infrastructure
- design and implement Sensor repository component of Sensor Manager
- begin exploration on the use of GMA as a generalized event handling system for Grid environments.
- determine scalability limitations to GMA and the event archive
- continued GGF leadership role on defining GMA protocol and API standards
- continued GGF leadership role on defining Network metrics for the Grid

Grid Project Integration

- deploy initial DMF on PPDG and DOE Science Grid testbeds
- continue to help Grid application and middleware developers use DMF/NetLogger for performance analyses

Year 3:

DMF

- design and implement event archive filtering
- research distributed querying of multiple event archives
- research caching / performance improvement strategies for the GMA
- design and implement event filters for GMA
- continued GGF leadership role on defining Network metrics for the Grid

Grid Project Integration

- assist in deployment of DMF components in several Grid projects
- continue to help Grid application and middleware developers use DMF/NetLogger for performance analyses

2.0 Bibliography of Literature

[1] Allcock B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S., “Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing”, http://www.globus.org/

[2] Bethel, W., Tierney, B., Lee, J., Gunter, D., Lau, S.,“Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization”, Proceeding of the IEEE Supercomputing 2000 Conference, Nov. 2000.

[3] Cancio, G., Fisher, S., Folkes, T., Giacomini, F., Hoschek, W., Kelsey, D., Tierney, B., “The DataGrid Architecture”, http://grid-atf.web.cern.ch/grid-atf/doc/architecture-2001-07-02.pdf

[4] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets”. Journal of Network and Computer Applications, 2000.

[5] Condor Project: http://www.cs.wisc.edu/condor/

[6] CORBA, “Systems Management: Event Management Service”, X/Open Document Number: P437, http://www.open-group.org/onlinepubs/008356299/

[7] Dinda, P., Lowecamp, B., “The Architecture of the REMOS System”, Proceedings of the IEEE High Performance Distributed Computing conference (HPDC-10), August 2001.

[8] ENABLE Project: http://www-didc.lbl.gov/ENABLE/

[9] Earth Systems Grid Project: http://www.scd.ucar.edu/css/esg/

[10] European Data Grid Project http://www.eu-datagrid.org/

[11] Fisk, M., Feng, W., “Dynamic Adjustment of TCP Window Sizes”, LANL Report: LA-UR 00-3221.

[12] Fitzgerald, S., I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, “A Directory Service for Configuring High-Performance Distributed Computations”. In Proceedings 6th IEEE Symposium on High Performance Distributed Computing, August 1997.

[13] Foster, I., and C. Kesselman, eds., The Grid: Blueprint for a New Computing Infrastructure, edited by Ian Foster and Carl Kesselman. Morgan Kaufmann, Pub. August 1998. ISBN 1-55860-475-8. http://www.mkp.com/books_catalog/1-55860-475-8.asp

[14] Grid Forum. The Grid Forum (www.Gridforum.org) is an informal consortium of institutions and individuals working on wide area computing and computational Grids: the technologies that underlie such activities as the NCSA Alliance's National Technology Grid, NPACI's Metasystems efforts, NASA's Information Power Grid, DOE ASCI's DISCOM program, and other activities worldwide.

[15] GGF Grid Performance Working Group, http://www-didc.lbl.gov/GridPerf/

[16] GGF Network Measurements Working Group: http://www.cs.wm.edu/~lowekamp/nmwg/

[17] GriPhyN Project: http://www.griphyn.org/

[18] Jin, G., Yang, G., Crowley, B., Agarwal, D., “Network Characterization Service”, Proceedings of the IEEE High Performance Distributed Computing conference, August 2001, http://www-didc.lbl.gov/NCS/

[19] iperf: http://dast.nlanr.net/Projects/Iperf/index.html

[20] Kalmady, R., Brian Tierney, “A Comparison of GSIFTP and RFIO on a WAN”, Cern Technical Report, March 2001. http://grid-data-management.web.cern.ch/grid-data-management/docs/GridFTP-rfio-report/

[21] Linux 2.4 autotuning: http://www.linuxhq.com/kernel/v2.4/doc/networking/ip-sysctl.txt.html

[22] Mathis, M., “Pushing Up Performance for Everyone”, Talk Slides, http://www.ncne.nlanr.net/news/workshop/1999/991205/Talks/mathis_991205_Pushing_Up_Performance/

[23] National Transparent Optical Network (NTON); http://www.ntonc.org/

[24] Net100: http://www.net100.org/

[25] NetLogger: http://www-didc.lbl.gov/NetLogger/

[26] Network Weather Service: http://nws.npaci.edu/NWS/

[27] pchar: http://www.employees.org/~bmah/Software/pchar/

[28] pipechar: http://www-didc.lbl.gov/~jin/network/net-tools.html

[29] Particle Physics Data Grid: http://www.ppdg.net/

[30] Peng, X, “Survey on Event Service”, http://www-unix.mcs.anl.gov/~peng/survey.html

[31] Ribler, R., Jeffrey S. Vetter, Huseyin Simitci, and Daniel A. Reed, “Autopilot: Adaptive Control of Distributed Applications,” Proceedings of the 7th IEEE Symposium on High-Performance Distributed Computing, Chicago, IL, July 1998.

[32] Self-Configuring Network Monitor: http://www-itg.lbl.gov/Net-Mon/Self-Config.html

[33] Schopf, J., “Ten Actions when Superscheduling”, GGF working draft, http://www.cs.northwestern.edu/~jms/sched-wg/WD/schedwd.8.5.pdf

[34] Semke, J. Mahdavi, M. Mathis, “Automatic TCP Buffer Tuning,” Computer Communication Review, ACM SIGCOMM, volume 28, number 4, Oct. 1998.

[35] Sivakumar, H, S. Bailey, R. L. Grossman, “PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks”, Proceedings of IEEE Supercomputing 2000, Nov., 2000. http://www.ncdm.uic.edu/html/psockets.html

[36] Smith, W., “A Framework for Control and Observation in Distributed Environments”, NAS Technical Report Number: NAS-01-006

[37] SuperNet Network Testbed Projects: http://www.ngi-supernet.org/

[38] Tierney, B., Gunter. D., Becla, J., Jacobsen, D., Quarrie, D.,“Using NetLogger for Distributed Systems Performance Analysis of the BaBar Data Analysis System”, Proceedings of Computers in High Energy Physics 2000 (CHEP 2000), Feb. 2000

[39] Tierney, B., Aydt, R., Gunter, D., Smith, W.,Taylor, V., Wolski, R., et al., “A Grid Monitoring Service Architecture”, Global Grid Forum White Paper, http://www-didc.blb.gov/GridPerf/

[40] Tierney, B. Lee, J., Crowley, B., Holding, M., Hylton, J., Drake, F., “A Network-Aware Distributed Storage Cache for Data Intensive Environments”, Proceeding of IEEE High Performance Distributed Computing conference (HPDC-8), August 1999, LBNL-42896.

[41] Tierney, B., Gunter, D., Evans, J., Lee, J., Stoufer, M., “Enabling Network-Aware Applications”, Proceedings of the IEEE High Performance Distributed Computing conference (HPDC-10), August 2001.

[42] Tierney, B., B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson, “A Monitoring Sensor Management System for Grid Environments”, Proceedings of the IEEE High Performance Distributed Computing conference (HPDC-9), August 2000, LBNL-45260.

[43] Tierney, B., W. Johnston, B. Crowley, G. Hoo, C. Brooks, D. Gunter “The NetLogger Methodology for High Performance Distributed Systems Performance Analysis”, Proceeding of IEEE High Performance Distributed Computing conference (HPDC-7), July 1998, LBNL-42611.

[44] Tierney, B. “TCP Tuning Guide for Distributed Application on Wide Area Networks”, Usenix ;login, Feb. 2001 (http://www-didc.lbl.gov/tcp-wan.html).

[45] “The WEB100 Project, Facilitating Effective and Transparent Network Use”, http://www.web100.org/

[46] Wolski, R., N. Spring, J. Hayes, “The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing,” Future Generation Computing Systems, 1999. http://nws.npaci.edu/NWS/.